


```

LL          IIIII
LL          IIIII
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LLLLLLLLLLL
LLLLLLLLLLL

            IIIII
            IIIII
            II
            II
            II
            II
            II
            II
            II
            II
            II
            II
            II
            II
            IIIII
            IIIII

SSSSSSSSS
SSSSSSSSS
SS
SS
SS
SS
        SSSSSS
        SSSSSS
                SS
                SS
                SS
                SS
SSSSSSSSS
SSSSSSSSS

```

```
0001 0 MODULE symbols (IDENT = 'V04-000') =
0002 1 BEGIN
0003 1
0004 1
0005 1 *****
0006 1 *
0007 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0008 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0009 1 *  ALL RIGHTS RESERVED.
0010 1 *
0011 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0012 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0013 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0014 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0015 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0016 1 *  TRANSFERRED.
0017 1 *
0018 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0019 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0020 1 *  CORPORATION.
0021 1 *
0022 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0023 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0024 1 *
0025 1 *
0026 1 *****
0027 1
0028 1
0029 1 ++
0030 1 FACILITY: Command language editor
0031 1
0032 1 ABSTRACT:
0033 1
0034 1 This facility is used to enhance the command language
0035 1 and allow user-written commands to be available in the
0036 1 language.
0037 1
0038 1 ENVIRONMENT:
0039 1
0040 1 VAX/VMS operating system. unprivileged user mode.
0041 1
0042 1 AUTHOR: Tim Halvorsen, Feb 1980
0043 1
0044 1 Modified by:
0045 1
0046 1 V03-001 DAS0001 David Solomon 03-Jul-1984
0047 1 Return success if trying to add a duplicate symbol (SPR 55578).
0048 1
0049 1 V002 DWT0006 David W. Thiel 10-Dec-1981
0050 1 Fix find_record to fail if asked for (n+1)st record.
0051 1
0052 1 V001 TMH0001 Tim Halvorsen 28-Mar-1981
0053 1 Clear upper word of descriptor passed to scan_symbols
0054 1 action routine, in case it uses the entire longword as
0055 1 the length rather than the lower word.
0056 1
0057 1 --
```

```
58 0058 1 |
59 0059 1 | Include files
60 0060 1 |
61 0061 1 |
62 0062 1 | LIBRARY 'SYSSLIBRARY:STARLET';          ! VAX/VMS common definitions
63 0063 1 |
64 0064 1 | ** REQUIRE 'SRCS:CLEDEF';                ! Utility definitions
65 0065 1 | ---
66 0066 1 |
67 0067 1 |         Require file for all modules in the command language editor
68 0068 1 |
69 0069 1 | IDENT V02-001
70 0070 1 |
71 0071 1 | ---
72 0072 1 |
73 0073 1 |
74 0074 1 | *****
75 0075 1 | *
76 0076 1 | *  COPYRIGHT (c) 1978, 1980, 1982 BY
77 0077 1 | *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
78 0078 1 | *  ALL RIGHTS RESERVED.
79 0079 1 | *
80 0080 1 | *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
81 0081 1 | *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
82 0082 1 | *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
83 0083 1 | *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
84 0084 1 | *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
85 0085 1 | *  TRANSFERRED.
86 0086 1 | *
87 0087 1 | *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
88 0088 1 | *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
89 0089 1 | *  CORPORATION.
90 0090 1 | *
91 0091 1 | *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
92 0092 1 | *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
93 0093 1 | *
94 0094 1 | *
95 0095 1 | *****
96 0096 1 |
97 0097 1 |
98 0098 1 | ++
99 0099 1 | FACILITY: Command language editor
100 0100 1 |
101 0101 1 | ABSTRACT:
102 0102 1 |
103 0103 1 |         This is the common require file for all modules in the
104 0104 1 |         command language editor.
105 0105 1 |
106 0106 1 | ENVIRONMENT:
107 0107 1 |
108 0108 1 |         VAX/VMS operating system, unprivileged user mode utility,
109 0109 1 |         operates at non-AST level.
110 0110 1 |
111 0111 1 | AUTHOR: Tim Halvorsen, Feb 1980
112 0112 1 |
113 0113 1 | MODIFIED BY:
114 0114 1 |
```

SYMBOLS
V04-000

^{H 5}
15-Sep-1984 23:52:01
14-Sep-1984 11:52:07

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[ACC.SRC]SYMBOLS.B32;1 Page 3 (1)

: 115
: 116
: 117

0115 1 |
0116 1 |
0117 1 |----

V02-001 BLS0089 Benn Schreiber
Add badvalue shared message

16-Oct-1981

```
119 0118 1
120 0119 1
121 0120 1 Define commonly used BLISS definitions
122 0121 1
123 0122 1
124 0123 1 ** REQUIRE 'LIB$:UTILDEF'; ! Commonly used BLISS definitions
125 0124 1 ---
126 0125 1
127 0126 1 Commonly used definitions for VMS modules written in BLISS
128 0127 1
129 0128 1 Version 'V03-000'
130 0129 1
131 0130 1 *****
132 0131 1 *
133 0132 1 * COPYRIGHT (c) 1978, 1980, 1982 BY
134 0133 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
135 0134 1 * ALL RIGHTS RESERVED.
136 0135 1 *
137 0136 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
138 0137 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
139 0138 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
140 0139 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
141 0140 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
142 0141 1 * TRANSFERRED.
143 0142 1 *
144 0143 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
145 0144 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
146 0145 1 * CORPORATION.
147 0146 1 *
148 0147 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
149 0148 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
150 0149 1 *
151 0150 1 *
152 0151 1 *****
153 0152 1
154 0153 1 ++
155 0154 1 ABSTRACT:
156 0155 1
157 0156 1 This is the common require file for any module written
158 0157 1 in BLISS
159 0158 1
160 0159 1 ENVIRONMENT:
161 0160 1
162 0161 1 VAX/VMS operating system.
163 0162 1
164 0163 1 AUTHOR: Tim Halvorsen, Feb 1980
165 0164 1
166 0165 1 MODIFIED BY:
167 0166 1
168 0167 1 ----
```

```
170 0168 1 |
171 0169 1 | Equated symbols
172 0170 1 |
173 0171 1 |
174 0172 1 | LITERAL
175 0173 1 | true = 1,
176 0174 1 | false = 0,
177 0175 1 | ok = 1,
178 0176 1 | error = 2,
179 0177 1 | quad = 8;
180 0178 1 |
181 0179 1 |
182 0180 1 | Define structure type for VMS structures
183 0181 1 |
184 0182 1 |
185 0183 1 | STRUCTURE
186 0184 1 | bblock [o, p, s, e; n] =
187 0185 1 | [n]
188 0186 1 | (bblock+o)<p,s,e>;
189 0187 1 |
190 0188 1 | MACRO
191 M 0189 1 | descriptor [] = ! Generate a static string descriptor
192 M 0190 1 | UPLIT (%CHARCOUNT (%STRING (%REMAINING))),
193 0191 1 | UPLIT BYTE (%STRING (%REMAINING))) %;
194 0192 1 |
195 0193 1 | MACRO
196 M 0194 1 | own_descriptor [] = ! Generate the actual static string descriptor
197 M 0195 1 | -BBLOCK [8] INITIAL(%CHARCOUNT(%STRING(%REMAINING))),
198 0196 1 | UPLIT BYTE (%STRING(%REMAINING))) %;
199 0197 1 |
200 0198 1 | MACRO
201 M 0199 1 | return_if_error(command) =
202 M 0200 1 | BEGIN
203 M 0201 1 | LOCAL
204 M 0202 1 | status;
205 M 0203 1 |
206 M 0204 1 | status = command;
207 M 0205 1 | IF NOT .status
208 M 0206 1 | THEN
209 M 0207 1 | RETURN .status;
210 0208 1 | END%;
211 0209 1 |
212 0210 1 | MACRO
213 M 0211 1 | signal_if_error(command) =
214 M 0212 1 | BEGIN
215 M 0213 1 | LOCAL
216 M 0214 1 | status;
217 M 0215 1 |
218 M 0216 1 | status = command;
219 M 0217 1 | IF NOT .status
220 M 0218 1 | THEN
221 M 0219 1 | BEGIN
222 M 0220 1 | SIGNAL(.status);
223 M 0221 1 | RETURN .status;
224 M 0222 1 | END;
225 0223 1 | END%;
226 0224 1 |
```

```
227 0225 1 |
228 0226 1 | Macro to implement a function (f) of the message severity level that
229 0227 1 | maps the various severity levels such that arithmetic comparisons of the
230 0228 1 | mapped values ( f(severity) ) yield a order of precedence that is
231 0229 1 | intuitively acceptable:
232 0230 1 |
233 0231 1 |
234 0232 1 | ERROR NAME OLDVAL NEWVAL
235 0233 1 |
236 0234 1 | F(SUCCESS) 1 --> 0
237 0235 1 | F(INFORMATIONAL) 3 --> 2
238 0236 1 | F(WARNING) 0 --> 1
239 0237 1 | F(ERROR) 2 --> 3
240 0238 1 | F(SEVERE_ERROR) 4 --> 7
241 0239 1 |
242 0240 1 |
243 0241 1 | MACRO
244 M 0242 1 | severity level (status) =
245 M 0243 1 | BEGIN
246 M 0244 1 | LOCAL code: BBLOCK [LONG];
247 M 0245 1 | code = status;
248 M 0246 1 | .code [sts$severity] - (4 * .code [sts$success]) + 3
249 0247 1 | END%;
250 0248 1 |
251 0249 1 | MACRO
252 M 0250 1 | cli$external(prefix) =
253 M 0251 1 | %IF %DECLARED(%QUOTE %QUOTE cli$prefix)
254 M 0252 1 | %THEN UNDECLARE %QUOTE %QUOTE cli$prefix; %FI
255 M 0253 1 | MACRO cli$prefix = prefix %QUOTE %;
256 M 0254 1 | EXTERNAL LITERAL
257 0255 1 | cli$external_loop(%REMAINING)%,
258 0256 1 |
259 M 0257 1 | cli$external_loop[name] =
260 0258 1 | %NAME(cli$prefix,name): UNSIGNED(8)%;
261 0259 1 |
262 0260 1 | MACRO
263 M 0261 1 | $external_literal(symbol) =
264 M 0262 1 | BEGIN
265 M 0263 1 | %IF NOT %DECLARED(symbol) %THEN EXTERNAL LITERAL symbol
266 M 0264 1 | %IF %LENGTH GTR 1 %THEN : %REMAINING %FI; %FI
267 M 0265 1 | symbol
268 0266 1 | END%;
269 0267 1 |
270 0268 1 | MACRO
271 M 0269 1 | $fab_dev(dev_bit) = ! Access FAB$L_DEV bits of FAB block
272 M 0270 1 | %BYTEOFFSET(fab$l_dev),
273 0271 1 | %BITPOSITION(%NAME('dev$v_',dev_bit)),1,0%;
274 0272 1 |
275 0273 1 |
276 0274 1 | $SHR_MESSAGES - a macro which defines facility-specific message codes
277 0275 1 | which are based on the system-wide shared message codes.
278 0276 1 |
279 0277 1 | $SHR_MESSAGES( name, code, (msg,severity), ... )
280 0278 1 |
281 0279 1 | where:
282 0280 1 | "name" is the name of the facility (e.g., COPY)
283 0281 1 | "code" is the corresponding facility code (e.g., 103)
```

```
.. 284      0282 1  |
.. 285      0283 1  |
.. 286      0284 1  |
.. 287      0285 1  |
.. 288      0286 1  |
.. 289      M 0287 1  |
.. 290      M 0288 1  |
.. 291      0289 1  |
.. 292      0290 1  |
.. 293      M 0291 1  |
.. 294      0292 1  |
.. 295      0293 1  |
.. 296      M 0294 1  |
.. 297      M 0295 1  |
.. 298      M 0296 1  |
.. 299      M 0297 1  |
.. 300      0298 1  |

      'msg' is the name of the shared message (e.g., BEGIN)
      'severity' is the desired message severity (e.g., 1, 0, 2)

      MACRO
      $SHR_MESSAGES( FACILITY_NAME, FACILITY_CODE ) =
      [ LITERAL
      $HR$MSG_IDS( FACILITY_NAME, FACILITY_CODE, %REMAINING ); %,
      $HR$MSG_IDS( FACILITY_NAME, FACILITY_CODE ) [ VALUE ] =
      $HR$MSG_CALC( FACILITY_NAME, FACILITY_CODE, %REMOVE(VALUE) ) %,
      $HR$MSG_CALC( FACILITY_NAME, FACILITY_CODE, MSG_ID, SEVERITY ) =
      %NAME(FACILITY_NAME, '$', MSG_ID) = %NAME('$HR$', MSG_ID) + FACILITY_CODE*65536 +
      %IF %DECLARED(%NAME('$ST$K', SEVERITY))
      %THEN %NAME('$ST$K', SEVERITY)
      %ELSE SEVERITY %FI-%;
```

```
.. 302 0299 1
.. 303 0300 1 !** REQUIRE 'LIB$CLIDEF.B32';          ! CLI command table definitions
.. 304 0301 1
.. 305 0302 1      Command language interpreter command table structures
.. 306 0303 1
.. 307 0304 1      IDENT          V03-003
.. 308 0305 1
.. 309 0306 1
.. 310 0307 1
.. 311 0308 1
.. 312 0309 1
.. 313 0310 1
.. 314 0311 1 *  COPYRIGHT (c) 1978, 1980, 1982 BY
.. 315 0312 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
.. 316 0313 1 *  ALL RIGHTS RESERVED.
.. 317 0314 1 *
.. 318 0315 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
.. 319 0316 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
.. 320 0317 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
.. 321 0318 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
.. 322 0319 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
.. 323 0320 1 *  TRANSFERRED.
.. 324 0321 1 *
.. 325 0322 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
.. 326 0323 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
.. 327 0324 1 *  CORPORATION.
.. 328 0325 1 *
.. 329 0326 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
.. 330 0327 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
.. 331 0328 1 *
.. 332 0329 1 *****
.. 333 0330 1
.. 334 0331 1
.. 335 0332 1 **
.. 336 0333 1 FACILITY: DCL & MCR Command language interpreters
.. 337 0334 1
.. 338 0335 1 ABSTRACT:
.. 339 0336 1
.. 340 0337 1      These are the command table structure definitions
.. 341 0338 1      which describe the generic command table format used
.. 342 0339 1      by the DCL and MCR command interpreters.
.. 343 0340 1
.. 344 0341 1 ENVIRONMENT:
.. 345 0342 1
.. 346 0343 1      VAX/VMS operating system. supervisor mode.
.. 347 0344 1
.. 348 0345 1 AUTHOR: Tim Halvorsen, Feb 1980
.. 349 0346 1
.. 350 0347 1 Modified by:
.. 351 0348 1
.. 352 0349 1      V03-003 PCG0005      Peter George      22-Nov-1982
.. 353 0350 1      Add INT_W_PMPTLEN and INT_L_PMPTADDR and remove
.. 354 0351 1      INT_L_PROMPT.
.. 355 0352 1
.. 356 0353 1      V03-002 PCG0004      Peter George      18-Oct-1982
.. 357 0354 1      Add VEC_C_PROMPTMAX, INT_L_PROMPT, and ENT_V_SPELL.
.. 358 0355 1
```

SYMBOLS
V04-000

N 5
15-Sep-1984 23:52:01
14-Sep-1984 11:52:07

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[ACC.SRC]SYMBOLS.B32;1
Page 9 (4)

359
360
361
0356 1
0357 1
0358 1

V03-001 PCG0003 Peter George 15-Jul-1982
Add INT data structure for CLISINTERFACE routines.

...	362	0359	1	
...	363	0360	1	
...	364	0361	1	
...	365	0362	1	
...	366	0363	1	
...	367	0364	1	
...	368	0365	1	
...	369	0366	1	
...	370	0367	1	

Note that the term 'SRO' stands for self-relative offset.
The actual address is computed by adding the signed contents
of the field to the address of the structure.

If the offset is zero, then there is no associated data.

```
371 0368 1
372 0369 1
373 0370 1
374 0371 1
375 0372 1
376 0373 1
377 0374 1
378 0375 1
379 0376 1
380 0377 1
381 0378 1
382 0379 1
383 0380 1
384 0381 1
385 0382 1
386 0383 1
387 0384 1
388 0385 1
389 0386 1
390 0387 1
391 0388 1
392 P 0389 1
393 P 0390 1
394 0391 1
395 0392 1
396 0393 1
397 0394 1
398 0395 1
399 0396 1
400 P 0397 1
401 P 0398 1
402 P 0399 1
403 0400 1
404 0401 1
405 0402 1
406 0403 1
407 0404 1
408 0405 1

!...$VECDEF
MACRO VEC_L_IMAGE_TBL = 0.0,32.0%; ! OFFSET TO IMAGE TABLE
MACRO VEC_L_PROMPT_TBL = 4.0,32.0%; ! OFFSET TO PROMPT TABLE
MACRO VEC_L_QUAL_TBL = 8.0,32.0%; ! OFFSET TO QUALIFIER TABLE
MACRO VEC_L_VERB_TBL = 12.0,32.0%; ! OFFSET TO BUILT-IN VERB TABLE
MACRO VEC_L_VERBEND = 16.0,32.0%; ! OFFSET TO END OF VERB_TBL
MACRO VEC_L_USRCMD = 20.0,32.0%; ! OFFSET TO USER VERB TABLE
MACRO VEC_L_USREND = 24.0,32.0%; ! OFFSET TO END OF USER VERB TABLE
MACRO VEC_L_COMDPTR = 28.0,32.0%; ! OFFSET TO BUILT-IN POINTER TABLE
MACRO VEC_L_USERPTR = 32.0,32.0%; ! OFFSET TO USER POINTER TABLE
MACRO VEC_L_FREE = 36.0,32.0%; ! OFFSET TO NEXT FREE BYTE
MACRO VEC_B_STRLVL = 40.0,8.0%; ! TABLE STRUCTURE LEVEL
LITERAL
SEQULST (VEC_C_GBL,0,1
      (STREVC,5) ! CURRENT STRUCTURE LEVEL
);
MACRO VEC_B_PROMPTMAX = 41.0,8.0%; ! MAXIMUM SIZE OF ANY PROMPT STRING
LITERAL VEC_C_LENGTH3 = 42;
LITERAL VEC_K_LENGTH3 = 42; ! LENGTH OF STR LEVEL 3 AND BEFORE VEC
MACRO VEC_B_CLT = 42.0,8.0%; ! CLT TYPE
LITERAL
SEQULST (VEC_C_GBL,0,1
      (DCL,0) ! TABLES ARE FOR DCL
      (MCR,1) ! TABLES ARE FOR MCR
);
MACRO VEC_W_SIZE = 44.0,16.0%; ! SIZE IN BYTES OF VECTOR AREA
LITERAL VEC_C_LENGTH = 60;
LITERAL VEC_K_LENGTH = 60; ! LENGTH OF VECTOR AREA
LITERAL VEC_C_PROMPTMAX = 32; ! MAXIMUM SIZE OF ANY PROMPT STRING
```

```
409 0406 1
410 0407 1
411 0408 1
412 0409 1
413 0410 1
414 0411 1
415 0412 1
416 0413 1
417 0414 1
418 0415 1
419 0416 1
420 0417 1
421 0418 1
422 0419 1
423 0420 1
424 0421 1
425 0422 1
426 0423 1
427 0424 1
428 0425 1
429 0426 1
430 0427 1
431 0428 1
432 0429 1
433 0430 1
434 0431 1
435 0432 1
436 0433 1
437 0434 1
438 0435 1
439 0436 1
440 0437 1
441 0438 1
442 0439 1
443 0440 1
444 0441 1
445 0442 1
446 0443 1
447 0444 1
448 0445 1
449 0446 1
450 0447 1
451 0448 1
452 0449 1
453 0450 1
454 0451 1
455 0452 1
456 0453 1
457 0454 1
458 0455 1
459 0456 1

      DEFINE COMMAND DESCRIPTOR BLOCK

      !...$CMDDEF
      MACRO      CMD_B_SIZE      = 0,0,8,0%;      ! SIZE OF COMMAND DESCRIPTOR BLOCK
      MACRO      CMD_B_VERBTYP   = 1,0,8,0%;      ! VERB GENERIC TYPE
      MACRO      CMD_B_PARMCNT   = 2,0,8,0%;      ! MIN/MAX PARAMETER COUNTS

      MACRO      CMD_V_MINPARM   = 2,0,4,0%;      ! MINIMUM NUMBER OF PARAMETERS REQUIRED
      MACRO      CMD_V_MAXPARM   = 2,4,4,0%;      ! MAXIMUM NUMBER OF PARAMETERS ALLOWED

      MACRO      CMD_B_FLAGS     = 3,0,8,0%;      ! COMMAND FLAGS

      MACRO      CMD_V_ABREV     = 3,0,1,0%;      ! COMMAND MAY BE ABBREVIATED NON-UNIQUELY
      LITERAL    CMD_M_ABREV     = 1,1,-1,0;      ! TO A SINGLE CHARACTER

      MACRO      CMD_V_NOSTAT    = 3,1,1,0%;      ! COMMAND DOES NOT RETURN VALID STATUS
      LITERAL    CMD_M_NOSTAT    = 1,2,-1,1;

      MACRO      CMD_V_FOREIGN   = 3,2,1,0%;      ! FOREIGN COMMAND - NO PARSING IS DONE
      LITERAL    CMD_M_FOREIGN   = 1,3,-1,2;

      MACRO      CMD_V_IMMED     = 3,3,1,0%;      ! COMMAND IS IMMEDIATELY DISPATCHED W/O PARSING
      LITERAL    CMD_M_IMMED     = 1,4,-1,3;

      MACRO      CMD_V_MCRPARSE  = 3,4,1,0%;      ! COMMAND IS MCR STYLE COMMAND (OUT=IN)
      LITERAL    CMD_M_MCRPARSE  = 1,5,-1,4;      ! (THIS FLAG ONLY EXAMINED BY MCR CLI)

      MACRO      CMD_W_IMAGE     = 4,0,16,1%;      ! SRO TO ASCII IMAGE NAME
      MACRO      CMD_W_QUALS     = 6,0,16,1%;      ! SRO TO FIRST NONPOSITIONAL ENTITY
      MACRO      CMD_W_PARMS     = 8,0,16,1%;      ! SRO TO FIRST POSITIONAL ENTITY
      MACRO      CMD_W_OUTPUTS   = 10,0,16,1%;     ! SRO TO LIST OF 'OUTPUT' ENTITIES
      MACRO      CMD_W_MutexSET  = 12,0,16,1%;     ! SRO TO MUTUAL EXCLUSION SET
      MACRO      CMD_W_IMPSET    = 14,0,16,1%;     ! SRO TO IMPLICATION SET

      LITERAL    CMD_C_LENGTH    = 16;
      LITERAL    CMD_K_LENGTH    = 16;      ! LENGTH OF FIXED PORTION

      OUTPUT LIST FORMAT:
      FIRST BYTE CONTAINS COUNT OF ENTRIES IN LIST EACH ENTRY IS ONE BYTE,
      SIGNED, DESCRIBING THAT 'OUTPUT NUMBER'. NEGATIVE VALUES INDICATE THE
      OUTPUT IS A PARAMETER AND THE ABS(VALUE) IS THE PARAMETER NUMBER.
      POSITIVE VALUES INDICATE THE OUTPUT IS A QUALIFIER AND THE VALUE IS A
      QUALIFIER NUMBER.

      QUAL IS (0:MAXQUALS-1),PARM IS (MAXQUALS:255)

      LITERAL    CMD_C_MAXPARMS  = 8;      ! MAXIMUM POSSIBLE PARAMETERS
      LITERAL    CMD_C_MAXQUALS  = 248;    ! MAXIMUM POSSIBLE QUALIFIERS (256-8)
```

460		0457	1	
461		0458	1	
462		0459	1	
463		0460	1	!!! DEFINE ENTITY DESCRIPTOR BLOCK
464		0461	1	
465		0462	1	
466		0463	1	!...SENTDEF
467		0464	1	MACRO ENT_B_NEXT = 0,0,8,0%; ! OFFSET TO NEXT BLOCK IN CHAIN
468		0465	1	MACRO ENT_B_SIZE = 1,0,8,0%; ! SIZE OF THIS BLOCK IN BYTES
469		0466	1	MACRO ENT_B_TYPE = 2,0,8,0%;
470		0467	1	LITERAL
471	P	0468	1	SEQULST (ENT C GBL 0,1
472	P	0469	1	, (PARAMETER,)
473	P	0470	1	, (QUALIFIER,)
474		0471	1	;:
475		0472	1	MACRO ENT_B_VALTYPE = 3,0,8,0%; ! TYPE OF VALUE
476		0473	1	LITERAL
477	P	0474	1	SEQULST (ENT C GBL 1,1
478	P	0475	1	, (INFILE,)
479	P	0476	1	, (OUTFILE,)
480	P	0477	1	, (NUMBER,)
481	P	0478	1	, (PRIVILEGE,)
482	P	0479	1	, (DATETIME,)
483	P	0480	1	, (PROTECTION,)
484	P	0481	1	, (PROCESS,)
485	P	0482	1	, (INLOG,)
486	P	0483	1	, (OUTLOG,)
487	P	0484	1	, (INSYM,)
488	P	0485	1	, (OUTSYM,)
489	P	0486	1	, (NODE,)
490	P	0487	1	, (DEVICE,)
491	P	0488	1	, (DIR,)
492	P	0489	1	, (UIC,)
493	P	0490	1	, (RESTOFLINE,)
494		0491	1	;:
495		0492	1	
496		0493	1	MACRO ENT_W_NAME = 4,0,16,1%; ! SRO TO ASCII ENTITY NAME (USER SPELLING)
497		0494	1	MACRO ENT_W_NUMBER = 4,0,16,0%; ! OR, PARAMETER NUMBER (POSITIONAL)
498		0495	1	MACRO ENT_W_LABEL = 6,0,16,1%; ! SRO TO ASCII ENTITY LABEL (FOR PGM USE)
499		0496	1	MACRO ENT_W_DEFVAL = 8,0,16,1%; ! SRO TO ASCII DEFAULT VALUE
500		0497	1	MACRO ENT_W_SYNTAX = 10,0,16,1%; ! SRO TO SYNTAX LIST
501		0498	1	MACRO ENT_W_KEYWORDS = 12,0,16,1%; ! SRO TO VALUE KEYWORD LIST
502		0499	1	! IF ZERO, ALL VALUES ARE LEGAL
503		0500	1	MACRO ENT_W_PROMPT = 14,0,16,1%; ! SRO TO VALUE PROMPT
504		0501	1	MACRO ENT_L_FLAGS = 16,0,32,0%; ! ENTITY FLAGS
505		0502	1	
506		0503	1	MACRO ENT_V_FILE = 16,0,1,0%; ! VALUE IS FILE SPECIFICATION
507		0504	1	LITERAL ENT_M_FILE = 1-1 - 1-0;
508		0505	1	MACRO ENT_V_VAL = 16,1,1,0%; ! CAN HAVE A VALUE
509		0506	1	LITERAL ENT_M_VAL = 1-2 - 1-1;
510		0507	1	MACRO ENT_V_NEG = 16,2,1,0%; ! VALUE CAN BE NEGATED
511		0508	1	LITERAL ENT_M_NEG = 1-3 - 1-2;
512		0509	1	MACRO ENT_V_DEFTTRUE = 16,3,1,0%; ! TRUE BY DEFAULT
513		0510	1	LITERAL ENT_M_DEFTTRUE = 1-4 - 1-3;
514		0511	1	MACRO ENT_V_BATCHDEF = 16,4,1,0%; ! PRESENT BY DEFAULT IF BATCH JOB
515		0512	1	LITERAL ENT_M_BATCHDEF = 1-5 - 1-4;
516		0513	1	MACRO ENT_V_VALREQ = 16,5,1,0%; ! VALUE IS REQUIRED

517	0514	1	LITERAL	ENT_M_VALREQ	= 1^6 - 1^5;	
518	0515	1	MACRO	ENT_V_LIST	= 16,6,1,0%;	! COMMA-SEPARATED LIST OF VALUES ALLOWED
519	0516	1	LITERAL	ENT_M_LIST	= 1^7 - 1^6;	
520	0517	1	MACRO	ENT_V_CONCAT	= 16,7,1,0%;	! CONCATENATED VALUES ALLOWED
521	0518	1	LITERAL	ENT_M_CONCAT	= 1^8 - 1^7;	
522	0519	1	MACRO	ENT_V_IMPCAT	= 16,8,1,0%;	! VALUES ARE IMPLICITLY CONCATENATED
523	0520	1	LITERAL	ENT_M_IMPCAT	= 1^9 - 1^8;	
524	0521	1	MACRO	ENT_V_VERB	= 16,9,1,0%;	! QUALIFIER CAN APPEAR ON COMMAND VERB
525	0522	1	LITERAL	ENT_M_VERB	= 1^10 - 1^9;	
526	0523	1	MACRO	ENT_V_PARM	= 16,10,1,0%;	! QUALIFIER CAN APPEAR ON PARAMETER
527	0524	1	LITERAL	ENT_M_PARM	= 1^11 - 1^10;	
528	0525	1	MACRO	ENT_V_MCROPTDLM	= 16,11,1,0%;	! VALUE DELIMITER IS OPTIONAL (MCR)
529	0526	1	LITERAL	ENT_M_MCROPTDLM	= 1^12 - 1^11;	
530	0527	1	MACRO	ENT_V_MCRIGNORE	= 16,12,1,0%;	! IGNORE THIS ENTITY BLOCK (MCR)
531	0528	1	LITERAL	ENT_M_MCRIGNORE	= 1^13 - 1^12;	
532	0529	1	MACRO	ENT_V_SPELL	= 16,13,1,0%;	! ONLY CHECK FIRST FOUR CHARS OF KEYWORD VALUES
533	0530	1	LITERAL	ENT_M_SPELL	= 1^14 - 1^13;	
534	0531	1				
535	0532	1	LITERAL	ENT_C_LENGTH	= 20;	
536	0533	1	LITERAL	ENT_K_LENGTH	= 20;	! LENGTH OF FIXED LENGTH PORTION
537	0534	1				

```
0535 1  
0536 1  
0537 1  
0538 1  
0539 1  
0540 1  
0541 1  
0542 1  
0543 1  
0544 1  
0545 1  
0546 1  
0547 1  
0548 1  
0549 1  
0550 1  
0551 1  
0552 1  
0553 1  
0554 1  
0555 1  
0556 1  
0557 1  
0558 1  
0559 1  
0560 1  
0561 1  
0562 1  
0563 1  
0564 1  
0565 1  
0566 1  
0567 1
```

DEFINE CHANGE LIST STRUCTURE

!...\$CHGDEF

MACRO	CHG_B_SIZE	= 0,0,8,0%;	! SIZE OF CHANGE LIST BLOCK
MACRO	CHG_B_FLAGS	= 1,0,8,0%;	! FLAGS
MACRO	CHG_V_IMAGE	= 1,0,1,0%;	! IMAGE CHANGE
LITERAL	CHG_M_IMAGE	= 1,1,1,0%;	
MACRO	CHG_V_PARMs	= 1,1,1,0%;	! PARAMETER(S) CHANGE
LITERAL	CHG_M_PARMs	= 1,2,1,1%;	
MACRO	CHG_V_QUALS	= 1,2,1,0%;	! QUALIFIER(S) CHANGE
LITERAL	CHG_M_QUALS	= 1,3,1,2%;	
MACRO	CHG_V_MCRIGNORE	= 1,3,1,0%;	! IGNORE IF CLI IS MCR
LITERAL	CHG_M_MCRIGNORE	= 1,4,1,3%;	
MACRO	CHG_W_IMAGE	= 2,0,16,1%;	! SRO TO NEW IMAGE
MACRO	CHG_B_PARMcnt	= 4,0,8,0%;	! MIN/MAX PARAMETER COUNTS
MACRO	CHG_V_MINPARM	= 4,0,4,0%;	! MINIMUM NUMBER OF PARAMETERS REQUIRED
MACRO	CHG_V_MAXPARM	= 4,4,4,0%;	! MAXIMUM NUMBER OF PARAMETERS ALLOWED
MACRO	CHG_W_PARMs	= 5,0,16,1%;	! SRO TO FIRST PARAMETER DESCRIPTOR
MACRO	CHG_W_QUALS	= 7,0,16,1%;	! SRO TO FIRST QUALIFIER DESCRIPTOR
LITERAL	CHG_C_LENGTH	= 9;	
LITERAL	CHG_K_LENGTH	= 9;	


```
0625 0622 1 |
0626 0623 1 |-----
0627 0624 1 |
0628 0625 1 |
0629 0626 1 |*****
0630 0627 1 |*
0631 0628 1 |*  COPYRIGHT (c) 1978, 1980, 1982 BY
0632 0629 1 |*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0633 0630 1 |*  ALL RIGHTS RESERVED.
0634 0631 1 |*
0635 0632 1 |*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0636 0633 1 |*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0637 0634 1 |*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0638 0635 1 |*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0639 0636 1 |*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0640 0637 1 |*  TRANSFERRED.
0641 0638 1 |*
0642 0639 1 |*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0643 0640 1 |*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0644 0641 1 |*  CORPORATION.
0645 0642 1 |*
0646 0643 1 |*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0647 0644 1 |*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0648 0645 1 |*
0649 0646 1 |*
0650 0647 1 |*****
0651 0648 1 |
0652 0649 1 |
0653 0650 1 |
0654 0651 1 |
0655 0652 1 |
0656 0653 1 |
0657 0654 1 |
0658 0655 1 |
0659 0656 1 |
0660 0657 1 |
0661 0658 1 |
0662 0659 1 |
0663 0660 1 |
0664 0661 1 |
0665 0662 1 |
0666 0663 1 |
0667 0664 1 |

Define symbol table entry
(Length is SYMSC_FIXEDLEN + .SYMSB_SYMLEN)

!...$SYMDEF
MACRO      SYMSL_LINK      = 0,0,32,0%;      ! LINK TO NEXT IN CHAIN
MACRO      SYMSL_VALUE     = 4,0,32,0%;      ! VALUE OF SYMBOL
MACRO      SYMSB_SYMLEN    = 8,0,8,0%;      ! LENGTH OF SYMBOL NAME
LITERAL    SYMSC_FIXEDLEN  = 9;
LITERAL    SYMSK_FIXEDLEN  = 9;
MACRO      SYMSI_SYMBOL    = 9,0,8,0%;      ! LENGTH OF FIXED PORTION OF ENTRY
MACRO      SYMSI_SYMBOL    = 9,0,8,0%;      ! SYMBOL NAME
```

```
.. 669      0665 1  |
.. 670      0666 1  | Table of contents
.. 671      0667 1  |
.. 672      0668 1  |
.. 673      0669 1  | FORWARD ROUTINE
.. 674      0670 1  |     add_record,      | Add record to linked list
.. 675      0671 1  |     find_record,    | Find record by number
.. 676      0672 1  |     delete_list,    | Deallocate entire record list
.. 677      0673 1  |     add_symbol,     | Add symbol to symbol table
.. 678      0674 1  |     lookup_symbol,  | Lookup symbol in symbol table
.. 679      0675 1  |     lookup_value,   | Lookup value in symbol table
.. 680      0676 1  |     scan_symbols,   | Scan all symbols in symbol table
.. 681      0677 1  |     delete_symbol,  | Delete symbol from symbol table
.. 682      0678 1  |     delete_symbols, | Delete entire symbol table
.. 683      0679 1  |     allocate,       | Allocate dynamic storage
.. 684      0680 1  |     deallocate;     | Deallocate dynamic storage
.. 685      0681 1  |
.. 686      0682 1  |
.. 687      0683 1  | | Storage definitions
.. 688      0684 1  | |
.. 689      0685 1  |
.. 690      0686 1  | GLOBAL
.. 691      0687 1  |     symbol_header:  VECTOR [64] | List of listheads for symbol tables
.. 692      0688 1  |                     INITIAL(REP 64 OF (0)); | Set all listheads empty
.. 693      0689 1  |
.. 694      0690 1  | |
.. 695      0691 1  | | External routines
.. 696      0692 1  | |
.. 697      0693 1  |
.. 698      0694 1  | EXTERNAL ROUTINE
.. 699      0695 1  |     lib$get_vm: ADDRESSING_MODE(GENERAL), | Allocate dynamic storage
.. 700      0696 1  |     lib$free_vm: ADDRESSING_MODE(GENERAL); | Deallocate dynamic storage
```

```
702 0697 1 GLOBAL ROUTINE add_record (listhead, address, length) =
703 0698
704 0699 ----
705 0700
706 0701 This routine adds a given data record to the
707 0702 end of a given linked list.
708 0703
709 0704 Inputs:
710 0705
711 0706 listhead = Address of listhead for list
712 0707 address = Address of data record
713 0708 length = Length of data record
714 0709
715 0710 Outputs:
716 0711
717 0712 routine = status (already signaled)
718 0713 ----
719 0714
720 0715 BEGIN
721 0716
722 0717 LOCAL
723 0718 new_entry: REF VECTOR, ! Address of newly allocated entry
724 0719 entry: REF VECTOR; ! Current entry address
725 0720
726 0721
727 P 0722 RETURN_IF_ERROR ! Allocate space; signal any error
728 0723 (allocate(.length+8,new_entry));
729 0724
730 0725 new_entry [1] = .length; ! Set length into entry
731 0726 CHSMOVE(.length, .address, new_entry [2]); ! Copy data into entry
732 0727
733 0728 entry = .listhead; ! Start at listhead itself
734 0729
735 0730 WHILE .entry [0] NEQ 0 ! While not end of list
736 0731 DO
737 0732 entry = .entry [0]; ! Link to next in chain
738 0733
739 0734 entry [0] = new_entry; ! set link of last entry to new one
740 0735 new_entry [0] = 0; ! and set new "end of list"
741 0736
742 0737 RETURN true;
743 0738
744 0739 1 END;
```

```
.TITLE SYMBOLS
.IDENT \V04-000\
.PSECT $GLOBALS,NOEXE,2
00000000# 00000 SYMBOL_HEADER::
.LONG 0[64]
.EXTRN LIB$GET_VM, LIB$FREE_VM
.PSECT $CODES,NOWRT,2
```

SYMBOLS
V04-000

15-Sep-1984 23:52:01
14-Sep-1984 11:52:07

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[ACC.SRC]SYMBOLS.B32;1 Page 20
(12)

			5E		007C	00000		.ENTRY	ADD_RECORD, Save R2,R3,R4,R5,R6	..	0697
					04	C2	00002	SUBL2	#4, SP	
					5E	DD	00005	PUSHL	SP	0723
	7E		OC	AC	08	C1	00007	ADDL3	#8, LENGTH, -(SP)	
			0000V	CF	02	FB	0000C	CALLS	#2, ALLOCATE	
				24	50	E9	00011	BLBC	STATUS, 3\$	
				56	6E	D0	00014	MOVL	NEW ENTRY, R6	0725
				A6	AC	D0	00017	MOVL	LENGTH, 4(R6)	
			04	BC	OC	AC	28	MOVC3	LENGTH, @ADDRESS, 8(R6)	0726
08	A6		08	50	OC	AC	D0	MOVL	LISTHEAD, ENTRY	0728
					04	AC	D0	MOVL	(ENTRY)	0730
					60	D5	00027	TSTL	2\$	
					05	13	00029	BEQL	(ENTRY), ENTRY	0732
				50	60	D0	0002B	MOVL	1\$	
					F7	11	0002E	BRB	R6, (ENTRY)	0734
				60	56	D0	00030	MOVL	(R6)	0735
					66	D4	00033	CLRL	#1, R0	0737
				50	01	D0	00035	MOVL		
					04	00038	3\$:	RET		0739

: Routine Size: 57 bytes, Routine Base: \$CODE\$ + 0000

```
746 0740 1 GLOBAL ROUTINE find_record (listhead, number, retadr) =
747 0741 1
748 0742 1 ---
749 0743 1
750 0744 1 This routine locates a given record of data by
751 0745 1 record number in any given list. The address
752 0746 1 returned is the address of the data itself.
753 0747 1
754 0748 1 Inputs:
755 0749 1
756 0750 1 listhead = Address of listhead of list
757 0751 1 number = Record number to find
758 0752 1 retadr = Address of longword to receive data address
759 0753 1
760 0754 1 Outputs:
761 0755 1
762 0756 1 routine = true if found, else false
763 0757 1 ---
764 0758 1
765 0759 2 BEGIN
766 0760 2
767 0761 2 LOCAL
768 0762 2 entry: REF VECTOR; ! Address of current entry
769 0763 2
770 0764 2 entry = ..listhead; ! Start at first entry
771 0765 2
772 0766 2 INCR i FROM 1 TO .number-1 ! Skip first number-1 entries
773 0767 2 DO
774 0768 2 BEGIN
775 0769 2 IF .entry EQL 0 ! If premature end of list,
776 0770 2 THEN
777 0771 2 RETURN false; ! return entry not found
778 0772 2 entry = .entry [0]; ! Skip to next entry in list
779 0773 2 END;
780 0774 2
781 0775 2 IF .entry EQL 0 ! End of list
782 0776 2 THEN
783 0777 2 RETURN false;
784 0778 2
785 0779 2 .retadr = entry [2]; ! Return address of data itself
786 0780 2 RETURN true; ! Return successful
787 0781 2
788 0782 1 END;
```

				0000 00000	.ENTRY FIND RECORD, Save nothing	0740
	51	04	BC	D0 00002	MOVL @LISTHEAD, ENTRY	0764
			50	D4 00006	CLRL 1	0766
			07	11 00008	BRB 2\$	
			51	D5 0000A 1\$:	TSTL ENTRY	0769
			15	13 0000C	BEQL 3\$	
	51		61	D0 0000E	MOVL (ENTRY), ENTRY	0772
F4	50	08	AC	F2 00011 2\$:	AOBLSS NUMBER, 1, 1\$	0766
			51	D5 00016	TSTL ENTRY	0775

SYMBOLS
V04-000

N 6
13-Sep-1984 23:52:01
14-Sep-1984 11:52:07

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[ACC.SRC]SYMBOLS.B32;1 Page 22
(13)

0C	BC	08	09	13	00018	BEQL	38		
	50		A1	9E	0001A	MOVAB	8(R1),	RETADR	0779
			01	D0	0001F	MOVL	#1, R0		0780
				04	00022	RET			
			50	D4	00023	CLRL	R0		0782
				04	00025	RET			

; Routine Size: 38 bytes. Routine Base: \$CODE\$ + 0039

```
790 0783 1 GLOBAL ROUTINE delete_list (listhead) =
791 0784
792 0785
793 0786
794 0787 This routine deallocates all storage associated
795 0788 with a given record list.
796 0789
797 0790 Inputs:
798 0791
799 0792 listhead = Address of listhead for list
800 0793
801 0794 Outputs:
802 0795
803 0796 None
804 0797
805 0798
806 0799 BEGIN
807 0800
808 0801 LOCAL
809 0802 entry: REF VECTOR; ! Address of current entry
810 0803
811 0804 entry = ..listhead; ! Start at first entry
812 0805
813 0806 WHILE .entry NEQ 0 ! For each entry in list,
814 0807 DO
815 0808 BEGIN
816 0809 LOCAL next_entry;
817 0810 next_entry = .entry [0]; ! Save pointer to next entry
818 0811 deallocate(.entry [1]+8, .entry); ! Deallocate memory for entry
819 0812 entry = .next_entry; ! Skip to next entry in list
820 0813 END;
821 0814
822 0815 .listhead = 0; ! Zero listhead
823 0816
824 0817 RETURN true; ! Success
825 0818
826 0819 1 END;
```

				000C 00000	.ENTRY DELETE LIST, Save R2,R3	0783
	52	04	BC	D0 00002	MOVL @LISTHEAD, ENTRY	0804
			14	13 00006	BEQL 2\$	0806
	53		62	D0 00008	MOVL (ENTRY), NEXT_ENTRY	0810
			52	DD 0000B	PUSHL ENTRY	0811
7E	04	A2	08	C1 0000D	ADDL3 #8, 4(ENTRY), -(SP)	
	0000V	CF	02	FB 00012	CALLS #2, DEALLOCATE	
		52	53	D0 00017	MOVL NEXT_ENTRY, ENTRY	0812
			EA	11 0001A	BRB 1\$	0806
		04	BC	D4 0001C	CLRL @LISTHEAD	0815
	50		01	D0 0001F	MOVL #1, R0	0817
				04 00022	RET	0819

; Routine Size: 35 bytes, Routine Base: \$CODE\$ + 005F

SYMBOLS
V04-000

^{6 7}
15-Sep-1984 23:52:01
14-Sep-1984 11:52:07

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[ACC.SRC]SYMBOLS.B32;1 Page 24
(14)

```
0820 1 GLOBAL ROUTINE add_symbol (table, name_desc, value) =
0821 1
0822 1 ---
0823 1
0824 1 This routine adds a given symbol name and value to
0825 1 the symbol table. The symbol table list is sorted
0826 1 by symbol name.
0827 1
0828 1 Inputs:
0829 1
0830 1 table = Symbol table index
0831 1 name_desc = Address of descriptor of symbol name
0832 1 value = Value to be assigned to the symbol
0833 1
0834 1 Outputs:
0835 1
0836 1 r0 = status (already signaled)
0837 1 ---
0838 1
0839 1 BEGIN
0840 1
0841 1 MAP
0842 1 name_desc: REF BBLOCK [DSC$K_S_BLN];! Address of name descriptor
0843 1
0844 1 LOCAL
0845 1 entry: REF BBLOCK,! Address of symbol table entry
0846 1 location: REF BBLOCK;! Address of closest symbol name
0847 1
0848 1 IF lookup_symbol (.table, .name_desc, location) ! If already in symbol table,
0849 1 THEN
0850 1 BEGIN
0851 1 ! SIGNAL(msg(dupsym), 1, .name_desc);! signal user with bad symbol
0852 1 RETURN 1;! return success
0853 1 END;
0854 1
0855 1
0856 P RETURN_IF_ERROR ! Allocate a symbol entry
0857 1 (allocate(sym$fixedlen+.name_desc [dsc$w_length],entry));
0858 1
0859 1 entry [sym$l_value] = .value;! Set value of symbol
0860 1 entry [sym$b_symlen] ! Set length of symbol
0861 1 = .name_desc [dsc$w_length];
0862 1
0863 1 CHSMOVE (.name_desc [dsc$w_length], ! Copy symbol
0864 1 .name_desc [dsc$a_pointer],
0865 1 entry [sym$t_symbol]);
0866 1
0867 1 entry [sym$l_link] ! Link into symbol table
0868 1 = .location [sym$l_link]; ! in sorted order
0869 1 location [sym$l_link] = .entry;
0870 1
0871 1 RETURN true;
0872 1
0873 1 END;
```

				007C	00000	.ENTRY	ADD_SYMBOL, Save R2,R3,R4,R5,R6	0820
	5E		08	C2	00002	SUBL2	#8, -SP	
			5E	DD	00005	PUSHL	SP	0848
	52	08	AC	DD	00007	MOVL	NAME_DESC, R2	
			52	DD	0000B	PUSHL	R2	
		04	AC	DD	0000D	PUSHL	TABLE	
0000V	CF		03	FB	00010	CALLS	#3, LOOKUP_SYMBOL	
	2C		50	E8	00015	BLBS	R0, 1\$	
		04	AE	9F	00018	PUSHAB	ENTRY	0857
	7E		62	3C	0001B	MOVZWL	(R2), -(SP)	
	6E		09	CO	0001E	ADDL2	#9, (SP)	
0000V	CF		02	FB	00021	CALLS	#2, ALLOCATE	
	1E		50	E9	00026	BLBC	STATUS, 2\$	
	56	04	AE	DD	00029	MOVL	ENTRY, R6	0859
	04	0C	AC	DD	0002D	MOVL	VALUE, 4(R6)	
	08		62	90	00032	MOVB	(R2), 8(R6)	0861
09	A6		62	28	0003A	MOVC3	(R2), @4(R2), 9(R6)	0865
	04		BE	DD	000	MOVL	@LOCATION, (R6)	0868
	66	00	56	DD	00	MOVL	R6, @LOCATION	0869
	BE		01	DD	00044	MOVL	#1, R0	0871
	50		04	DD	00047	RET		0873
					1\$:			
					2\$:			

; Routine Size: 72 bytes. Routine Base: \$CODE\$ + 0082

```
0883 0874 1 GLOBAL ROUTINE lookup_symbol (table, name_desc, value) =
0884 0875 1
0885 0876 1 ---
0886 0877 1
0887 0878 1 This routine looks up a given symbol in the symbol
0888 0879 1 table and returns the value associated with it.
0889 0880 1 If the symbol is not found, then the address of the
0890 0881 1 last entry preceeding the symbol in collation
0891 0882 1 sequence is returned instead.
0892 0883 1
0893 0884 1 Inputs:
0894 0885 1
0895 0886 1 table = Symbol table index (1-n)
0896 0887 1 name_desc = Descriptor of desired symbol name
0897 0888 1 value = Address of longword to receive value if found
0898 0889 1
0899 0890 1 Outputs:
0900 0891 1
0901 0892 1 value = Value of symbol if found
0902 0893 1 value = Address of previous entry if not found
0903 0894 1 r0 = status
0904 0895 1 ---
0905 0896 1
0906 0897 1 BEGIN
0907 0898 1
0908 0899 1 MAP
0909 0900 1 name_desc: REF BBLOCK [DSC$K_S_BLN]; ! Address of descriptor
0910 0901 1
0911 0902 1 LOCAL
0912 0903 1 ptr: REF BBLOCK; ! Pointer into list
0913 0904 1
0914 0905 1 ptr = symbol_header [.table] - $BYTEOFFSET(sym$l_link); ! Start at listhead
0915 0906 1 .value = .ptr;
0916 0907 1
0917 0908 1 WHILE (ptr = .ptr [sym$l_link]) NEQ 0 ! Until end of list
0918 0909 1 DO
0919 0910 1 CASE CH$COMPARE(.ptr [sym$b_symlen], ptr [sym$t_symbol],
0920 0911 1 name_desc [dsc$b_length], .name_desc [dsc$a_pointer])
0921 0912 1 FROM -1 TO 1 OF SET
0922 0913 1 [-1]: .value = .ptr; ! Table symbol < user symbol
0923 0914 1 [0]: BEGIN ! Table symbol = user symbol
0924 0915 1 .value = .ptr [sym$l_value]; ! Return value of symbol
0925 0916 1 RETURN true; ! and exit successful
0926 0917 1 END;
0927 0918 1 [1]: RETURN false; ! Table symbol > user symbol
0928 0919 1 TES;
0929 0920 1
0930 0921 1 RETURN false; ! return symbol not found
0931 0922 1
0932 0923 1 END;
```

50 04 005C 00000
AC DO 00002

.ENTRY LOOKUP_SYMBOL, Save R2,R3,R4,R6
MOVL TABLE, -R0

: 0874
: 0905

SYMBOLS
V04-000

6 7
13-Sep-1984 23:52:01
14-Sep-1984 11:52:07

VAX-11 BLISS-32 V4.0-742
DISKSVMSMASTER:[ACC.SRC]SYMBOLS.B32;1 Page 28
(16)

			0C	54	0000'CF40	DE	00006	MOVAL	SYMBOL HEADER[R0], PTR		
				BC		D0	0000C	MOVL	PTR, @VALUE		0906
				56	08	AC	00010	MOVL	NAME_DESC, R6		0911
				54		D0	00014	MOVL	(PTR), PTR		0908
						20	13	BEQL	3\$		
				50	08	A4	9A	MOVZBL	8(PTR), R0		0910
08	BC	00	09	A4		50	2D	CMPC5	R0, 9(PTR), #0, @NAME_DESC, @4(R6)		
					04	B6					
						11	1A	BGTRU	3\$		
						06	1E	BGEQU	2\$		
			0C	BC		54	D0	MOVL	PTR, @VALUE		0913
						E4	11	BRB	1\$		
			0C	BC	04	A4	D0	MOVL	4(PTR), @VALUE		0915
				50		01	D0	MOVL	#1, R0		0916
							04	RET			
						50	D4	CLRL	R0		0923
						04	0003B	RET			

; Routine Size: 60 bytes, Routine Base: \$CODE\$ + 00CA

```
0934 GLOBAL ROUTINE lookup_value (table, value, retdesc) =
0935
0936 ---
0937 This routine locates the first occurrence of a symbol
0938 containing the specified value and returns a descriptor
0939 of the symbol associated with the value.
0940
0941 Inputs:
0942     table = Symbol table index (1-n)
0943     value = Value to be looked up
0944     retdesc = Address of quadword to receive descriptor
0945
0946 Outputs:
0947     routine = status
0948 ---
0949 BEGIN
0950 MAP
0951     retdesc:    REF VECTOR;           ! Address of descriptor
0952
0953 LOCAL
0954     ptr:        REF BBLOCK;          ! Pointer into list
0955
0956 ptr = .symbol_header [.table];      ! Start at first entry
0957
0958 WHILE .ptr NEQ 0                    ! Until end of list
0959 DO
0960     BEGIN
0961     IF .ptr [sym$l_value] EQL .value ! If value matches,
0962     THEN
0963         BEGIN
0964         retdesc [0] = .ptr [sym$b_symlen]; ! Return descriptor of name
0965         retdesc [1] = ptr [sym$t_symbol];
0966         RETURN true;                      ! and exit successful
0967         END;
0968     ptr = .ptr [sym$l_link];            ! If no match, go to next entry
0969     END;
0970
0971 RETURN false;                       ! return symbol not found
0972
0973 END;
```

				0000 0000	.ENTRY	LOOKUP_VALUE, Save nothing	0924
	50	04	AC	DO 00002	MOVL	TABLE, R0	0951
	51	0000'CF40	DO	00006	MOVL	SYMBOL_HEADER[R0], PTR	
			1D	13 0000C	BEQL	3\$	0953
08	AC	04	A1	D1 0000E	CMPL	4(PTR), VALUE	0956
			11	12 00013	BNEQ	2\$	
	50	0C	AC	DO 00015	MOVL	RETDESC, R0	0959

SYMBOLS
V04-000

1-7
13-Sep-1984 23:52:01
14-Sep-1984 11:52:07

VAX-11 B11ss-32 V4.0-742
DISK\$VMSMASTER:[ACC.SRC]SYMBOLS.B32;1
Page 30
(17)

04	60	08	A1	9A	00019	MOVZBL	8(PTR), (R0)	:		
	A0	09	A1	9E	0001D	MOVAB	9(R1), 4(R0)	:	0960	
	50		01	D0	00022	MOVL	#1, R0	:	0961	
				04	00025	RET		:		
	51		61	D0	00026	28:	MOVL	(PTR), PTR	:	0963
			E1	11	00029	BRB	1\$:	0953	
			50	D4	0002B	3\$:	CLRL	R0	:	0966
				04	0002D	RET		:	0968	

: Routine Size: 46 bytes. Routine Base: \$CODE\$ + 0106

```
0969 1 GLOBAL ROUTINE scan_symbols (table, action_routine) =
0970 1
0971 1 ---
0972 1
0973 1 This routine calls a specified action routine for
0974 1 each symbol in the specified symbol table.
0975 1
0976 1 Inputs:
0977 1
0978 1 table = Symbol table index (1-n)
0979 1 action_routine = Address of action routine to call
0980 1 with the following argument list:
0981 1 1) Address of descriptor of symbol name
0982 1 2) Value associated with symbol
0983 1
0984 1 Outputs:
0985 1
0986 1 The status of the last action routine executed is returned.
0987 1 ---
0988 1
1000 1 BEGIN
1001 1
1002 1 LOCAL
1003 1 status, ! Catch-all status return bucket
1004 1 desc: VECTOR [2], ! Descriptor of symbol name
1005 1 ptr: REF BBLOCK; ! Address of current symbol entry
1006 1
1007 1 ptr = .symbol_header [.table]; ! Start at first entry
1008 1
1009 1 WHILE .ptr NEQ 0 ! Until end of list,
1010 1 DO ! Setup descriptor of name
1011 1 BEGIN
1012 1 desc [0] = .ptr [sym$b_symlen];
1013 1 desc [1] = ptr [sym$t_symbol];
1014 1 status = (.action_routine)(desc, .ptr [sym$l_value]); ! Call action routine
1015 1 IF NOT .status THEN EXITLOOP; ! If failed, exit unsuccessful
1016 1 ptr = .ptr [sym$l_link]; ! Skip to next in chain
1017 1 END;
1018 1
1019 1 RETURN .status; ! return successful
1020 1
1021 1 END;
```

```
0004 00000
04 08 C2 00002
50 AC D0 00005
52 0000'CF40 D0 00009
08 1B 13 0000F 1$:
04 6E 08 A2 9A 00011
AE 09 A2 9E 00015
04 04 A2 DD 0001A
08 04 AE 9F 0001D
BC 02 FB 00020

.ENTRY SCAN_SYMBOLS, Save R2
SUBL2 #8, SP
MOVL TABLE, R0
MOVL SYMBOL_HEADER[R0], PTR
BEQL 2$,
MOVZBL 8(PTR), DESC
MOVAB 9(R2), DESC+4
PUSHL 4(PTR)
PUSHAB DESC
CALLS #2, @ACTION_ROUTINE
```

```
0969
0996
0998
1001
1002
1003
```

SYMBOLS
V04-000

15-Sep-1984 23:52:01 VAX-11 Bliss-32 V4.0-742 Page 32
14-Sep-1984 11:52:07 DISK\$VMSMASTER:[ACC.SRC]SYMBOLS.B32;1 (18)

05	50	E9	00024	BLBC	STATUS, 2\$:	1004
52	62	D0	00027	MOVL	(PTR), PTR	:	1005
	E3	11	0002A	BRB	1\$:	0998
	04	0002C	2\$:	RET		:	1010

; Routine Size: 45 bytes, Routine Base: \$CODE\$ + 0134

```
1023 1011 GLOBAL ROUTINE delete_symbol (table, name_desc) =
1024 1012
1025 1013
1026 1014
1027 1015 This routine deletes a given symbol from the symbol
1028 1016 table.
1029 1017
1030 1018 Inputs:
1031 1019
1032 1020 table = Symbol table index (1-n)
1033 1021 name_desc = Descriptor of symbol name to be deleted
1034 1022
1035 1023 Outputs:
1036 1024
1037 1025 r0 = true if deleted, false if not found
1038 1026
1039 1027
1040 1028 BEGIN
1041 1029
1042 1030 MAP
1043 1031 name_desc: REF BBLOCK [DSC$K_S_BLN]; ! Address of descriptor
1044 1032
1045 1033 LOCAL
1046 1034 prev: REF BBLOCK, ! Pointer to previous symbol
1047 1035 ptr: REF BBLOCK; ! Pointer into list
1048 1036
1049 1037 ptr = symbol_header [.table] - $BYTEOFFSET(sym$l_link); ! Start at listhead
1050 1038 prev = .ptr; ! Ditto
1051 1039
1052 1040 WHILE (ptr = .ptr [sym$l_link]) NEQ 0 ! Until end of list
1053 1041 DO
1054 1042 CASE CH$COMPARE(.ptr [sym$b_symlen], ptr [sym$t_symbol],
1055 1043 name_desc [dsc$a_length], .name_desc [dsc$a_pointer])
1056 1044 FROM -1 TO 1 OF SET
1057 1045 [-1]: prev = .ptr; ! Table symbol < user symbol
1058 1046 [0]: BEGIN ! Table symbol = user symbol
1059 1047 prev [sym$l_link] = .ptr [sym$l_link]; ! Delink it
1060 1048 RETURN deallocate (sym$c_fixedlen+.ptr[sym$b_symlen], .ptr); ! free VM
1061 1049 END;
1062 1050 [1]: RETURN false; ! Table symbol > user symbol
1063 1051 TES;
1064 1052
1065 1053
1066 1054 RETURN false; ! return symbol not found
1067 1055
1068 1056 END;
```

50	04	AC	DO	00002	.ENTRY	DELETE_SYMBOL, Save R2,R3,R4,R6,R7	1011
54	0000'CF	40	DE	00006	MOVL	TABLE, R0	1037
57		54	DO	0000C	MOVAL	SYMBOL_HEADER[R0], PTR	
56	08	AC	DO	0000F	MOVL	PTR, PREV	1038
54		64	DO	00013	MOVL	NAME_DESC, R6	1043
				18:	MOVL	(PTR), PTR	1040

SYMBOLS
V04-000

M 7
15-Sep-1984 23:52:01
14-Sep-1984 11:52:07

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[ACC.SRC]SYMBOLS.B32;1 Page 34
(19)

08	BC	00	09	50	08	28	13	00016	BEQL	3\$		
				A4		A4	9A	00018	MOVZBL	8(PTR), R0		1042
					04	50	2D	0001C	CMPCS	R0, 9(PTR), #0, @NAME_DESC, @4(R6)		
						B6	1A	00023				
						19	1A	00025	BGTRU	3\$		
						05	1E	00027	BGEQU	2\$		
				57		54	D0	00029	MOVL	PTR, PREV		1045
						E5	11	0002C	BRB	1\$		
				67		64	D0	0002E	2\$: MOVL	(PTR), (PREV)		1047
						54	DD	00031	PUSHL	PTR		1048
				7E	08	A4	9A	00033	MOVZBL	8(PTR), -(SP)		
				6E		09	C0	00037	ADDL2	#9, (SP)		
				CF		02	FB	0003A	CALLS	#2, DEALLOCATE		
						04	04	0003F	RET			
						50	D4	00040	3\$: CLRL	R0		1056
						04	04	00042	RET			

; Routine Size: 67 bytes, Routine Base: \$CODE\$ + 0161

```
1070 1 GLOBAL ROUTINE delete_symbols (table) =
1071 1058 1
1072 1059 1
1073 1060 1
1074 1061 1 This routine deallocates all symbols in a specified
1075 1062 1 symbol table.
1076 1063 1
1077 1064 1 Inputs:
1078 1065 1
1079 1066 1 table = Symbol table index (1-n)
1080 1067 1
1081 1068 1 Outputs:
1082 1069 1
1083 1070 1 None
1084 1071 1
1085 1072 1
1086 1073 1 BEGIN
1087 1074 1
1088 1075 1 LOCAL
1089 1076 1 ptr: REF BBLOCK; ! Address of current entry
1090 1077 1
1091 1078 1 ptr = .symbol_header [.table]; ! Start at first entry
1092 1079 1
1093 1080 1 WHILE .ptr NEQ 0 ! Until end of list,
1094 1081 1 DO
1095 1082 1 BEGIN
1096 1083 1 LOCAL next_entry;
1097 1084 1 next_entry = .ptr [sym$1_link]; ! Save pointer to next entry
1098 1085 1 deallocate(sym$1_fixedlen+.ptr [sym$1_symlen], .ptr); ! Deallocate entry
1099 1086 1 ptr = .next_entry; ! Point to next entry in list
1100 1087 1 END;
1101 1088 1
1102 1089 1 symbol_header [.table] = 0; ! Zero listhead
1103 1090 1
1104 1091 1 RETURN true;
1105 1092 1
1106 1093 1 END;
```

			001C 00000	.ENTRY	DELETE_SYMBOLS, Save R2,R3,R4	1057
52	04	AC	D0 00002	MOVL	TABLE, R2	1078
53	0000'CF	42	D0 00006	MOVL	SYMBOL_HEADER[R2], PTR	
		16	13 0000C	BEQL	2\$	1080
54		63	D0 0000E	MOVL	(PTR), NEXT_ENTRY	1084
		53	DD 00011	PUSHL	PTR	1085
7E	08	A3	9A 00013	MOVZBL	8(PTR), -(SP)	
6E		09	C0 00017	ADDL2	#9, (SP)	
0000V		02	FB 0001A	CALLS	#2, DEALLOCATE	
53		54	D0 0001F	MOVL	NEXT_ENTRY, PTR	1086
		E8	11 00022	BRB	1\$	1080
	0000'CF	42	D4 00024	CLRL	SYMBOL_HEADER[R2]	1089
50		01	D0 00029	MOVL	#1, R0	1091
			04 0002C	RET		1093

SYMBOLS
V04-000

15-Sep-1984 23:52:01
14-Sep-1984 11:52:07

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[ACC.SRC]SYMBOLS.B32;1 Page 36 (20)

; Routine Size: 45 bytes, Routine Base: \$CODE\$ + 01A4

```
1108 1094 1 GLOBAL ROUTINE allocate (bytes, address) =
1109 1095 1
1110 1096 1 ---
1111 1097 1
1112 1098 1 Allocate dynamic storage and return the address.
1113 1099 1 If an error occurs, the error is signaled.
1114 1100 1
1115 1101 1 Inputs:
1116 1102 1
1117 1103 1 bytes = Number of bytes to allocate
1118 1104 1 address = Longword to receive address of storage
1119 1105 1
1120 1106 1 Outputs:
1121 1107 1
1122 1108 1 address = Address of storage
1123 1109 1 ---
1124 1110 1
1125 1111 1 BEGIN
1126 1112 1
1127 1113 1 LOCAL
1128 1114 1 status;
1129 1115 1
1130 1116 1 status = lib$get_vm(bytes,.address);
1131 1117 1
1132 1118 1 IF NOT .status ! if unsuccessful,
1133 1119 1 THEN
1134 1120 1 SIGNAL(.status); ! then signal the error
1135 1121 1
1136 1122 1 RETURN .status; ! return with status;
1137 1123 1
1138 1124 1 END;
```

			0004 00000	.ENTRY	ALLOCATE, Save R2	: 1094
		08	AC DD 00002	PUSHL	ADDRESS	: 1116
		04	AC 9F 00005	PUSHAB	BYTES	:
00000000G	00		02 FB 00008	CALLS	#2, LIB\$GET_VM	:
	52		50 D0 0000F	MOVL	R0, STATUS	:
	09		52 E8 00012	BLBS	STATUS, 1\$: 1118
			52 DD 00015	PUSHL	STATUS	: 1120
00000000G	00		01 FB 00017	CALLS	#1, LIB\$SIGNAL	:
	50		52 D0 0001E 1\$:	MOVL	STATUS, R0	: 1122
			04 00021	RET		: 1124

; Routine Size: 34 bytes, Routine Base: \$CODE\$ + 01D1

```
1140 1125 1 GLOBAL ROUTINE deallocate (bytes, address) =
1141 1126 1
1142 1127 1 ---
1143 1128 1
1144 1129 1     Deallocate dynamic storage.
1145 1130 1     If an error occurs, the error is signaled.
1146 1131 1
1147 1132 1 Inputs:
1148 1133 1
1149 1134 1     bytes = Number of bytes to deallocate
1150 1135 1     address = Address of storage to deallocate
1151 1136 1
1152 1137 1 Outputs:
1153 1138 1
1154 1139 1     None
1155 1140 1 ---
1156 1141 1
1157 1142 1 BEGIN
1158 1143 1
1159 1144 1 LOCAL
1160 1145 1     status;
1161 1146 1
1162 1147 1 status = lib$free_vm(bytes,address);
1163 1148 1
1164 1149 1 IF NOT .status          ! if unsuccessful,
1165 1150 1 THEN
1166 1151 1     SIGNAL(.status);    ! then signal the error
1167 1152 1
1168 1153 1 RETURN .status;         ! return with status;
1169 1154 1
1170 1155 1 END;
```

		08	0004	00000	.ENTRY	DEALLOCATE, Save R2	: 1125
		04	AC	9F 00002	PUSHAB	ADDRESS	: 1147
			AC	9F 00005	PUSHAB	BYTES	:
00000000G	00		02	FB 00008	CALLS	#2, LIB\$FREE_VM	:
	52		50	D0 0000F	MOVL	R0, STATUS	:
	09		52	E8 00012	BLBS	STATUS, 1\$: 1149
			52	DD 00015	PUSHL	STATUS	: 1151
00000000G	00		01	FB 00017	CALLS	#1, LIB\$SIGNAL	:
	50		52	D0 0001E	MOVL	STATUS, R0	: 1153
			04	00021	RET		: 1155

; Routine Size: 34 bytes, Routine Base: \$CODE\$ + 0153

SYMBOLS
V04-000

E 8
15-Sep-1984 23:52:01
14-Sep-1984 11:52:07

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[CACC.SRC]SYMBOLS.B32;1 Page 39 (23)

: 1172
: 1173
1156 1 END
1157 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$GLOBALS	256	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	533	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	22	0	581	00:01.0

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:SYMBOLS/OBJ=OBJ\$:SYMBOLS MSRC\$:SYMBOLS/UPDATE=(ENH\$:SYMBOLS)

: Size: 533 code + 256 data bytes
: Run Time: 00:19.7
: Elapsed Time: 00:49.6
: Lines/CPU Min: 3529
: Lexemes/CPU-Min: 22581
: Memory Used: 90 pages
: Compilation Complete

0002

**DIGITAL
CONFIDE**